



Unit 3: Brightness, if and while with the TI-Innovator™ Hub

Skill Builder 1: Measuring Light

In this lesson, you will take control of the brightness sensor on the TI-Innovator Hub and learn how to make use of its information.

Objectives:

- Read the brightness sensor
- Set the range of the brightness sensor
- Monitor the brightness sensor
- Control a light with the brightness sensor

Unlike the light, color, and sound features of the TI-Innovator Hub, the brightness sensor is an *Input device* rather than an *Output device*. A program can obtain information *from* the brightness sensor and take actions based on that numeric value. You can either work with the default brightness values or you can set the range of values with a special **brightness.range()** function.

The brightness sensor is clearly labeled on one end of the TI-Innovator Hub.



Teacher Tip: This first project will simply read and display the brightness values on the screen.

1. Begin a new Python Hub Project, and start with the three statements

```

cls()
text_at(13,"Press [esc] to end","center")
while get_key() != "esc":
    block

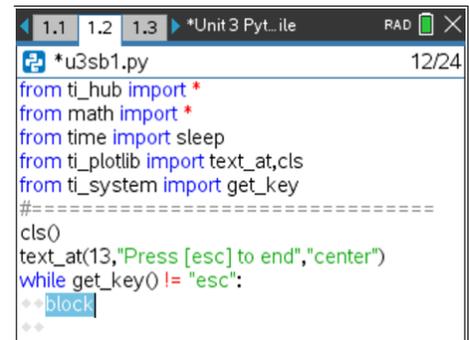
```

all found at **menu > TI Hub > Commands**.

cls() clears the screen.

text_at(13, ...) prints the message at the bottom center of the screen.

When running the program, press **esc** to end.



2. In the **while** block, use two statements: one to read the brightness and one to display the value.

Assign **b** to read the brightness measurement

```

b = brightness.measurement()

```

found on **menu > TI Hub > Hub Built-in Devices > Brightness Input**.

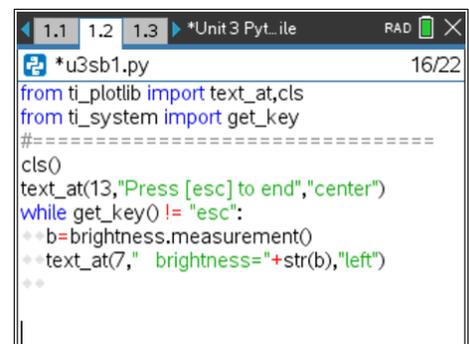
Add:

```

text_at(7, "brightness = " + str(b), "left")

```

text_at(7... is the vertical middle of the screen.





3. About "brightness = " + str(b):

str(b) (found on **menu > Built-ins > Type**) converts the numeric value of **b** into a string because **text_at()** can only display text (characters), not values of numeric variables.

The **+** sign combines the word "brightness = " with the *string* value of **b**. This type of 'addition of strings' is called *concatenation*.

Alignment "**left**" is better than "**center**" in this case. If you prefer to use "**center**", then new data might not completely erase old data since the line will vary in length. You can move the text closer to the center by adding spaces before " brightness = " (inside the quotes).

4. Run the program now to see the effect on the screen. You should see something like this, and your brightness value will be changing.

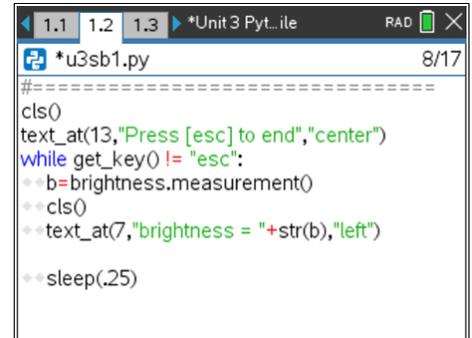
```
# b=brightness.measurement()
text_at(7,"Brightness="+str(b),"left")
```

```
Brightness = 13.5436
```



5. Slow the display down a bit by adding a **sleep()** statement right after the **text_at()** statement. Be sure it is indented to match the other statements in the **while** block.

Run the program again and determine the lowest and highest values that the sensor currently delivers by changing the light intensity hitting the sensor.



Teacher Tip: The brightness default range is [0,100]. A smartphone 'flashlight' makes a good light source. A brightness value of 0 can be hard to achieve.

6. You can set the range of values that the brightness sensor delivers using the statement

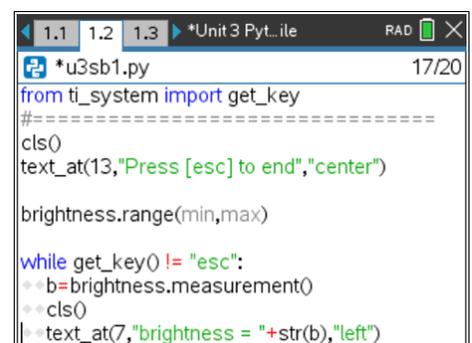
brightness.range(min, max)

found on **menu > TI Hub > Hub Built-in Devices > Brightness Input**.

Be sure to place this statement *before* the while loop. Use any values for *min* and *max* but be sure that *min* < *max*.

Change the range and run the program again to observe the values produced. You now have a custom digital light meter.

Why is this important? Check out the next few lessons...





10 Minutes of Code - Python

TI-NSPIRE™ CX II WITH THE TI-INNOVATOR™ HUB

UNIT 3: SKILL BUILDER 1

TEACHER NOTES

Teacher Tip: Setting the range makes working with other TI-Innovator Hub devices like sound and color a lot easier because there is no need to 'convert' from one scale to another. This is explored in the following lessons.

It is possible to re-define the brightness range within a program, but this is rarely necessary.